Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πολιτικές αντικατάστασης cache για blockchain

Διπλωματική Εργασία

Ιωάννης Ντζιός

Επιβλέπων: Δημήτριος Κατσαρός
Επίκουρος Καθηγητής Π.Θ.

Σεπτέμβριος, 2019

University of Thessaly
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Cache replacement policies for blockchain

Diploma Thesis

Ioannis Ntzios

Supervisor: Dimitrios Katsaros
Assistant Professor UTH

September, 2019

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα επίκουρο καθηγητή κύριο Δημήτριο Κατσαρό για την πολύτιμη βοήθεια του καθώς και καθοδήγηση του τοσο για την εκπόνηση της παρούσας διπλωματικής εργασίας, όσο και κατα τη διάρκεια των σπουδών μου. Θα ήθελα επίσης να ευχαριστήσω όλους του καθηγητές του τμήματος τους οποίους είχα την τιμή να γνωρίσω τα τελευταία χρόνια. Πάνω απ'όλα να πω και ένα μεγάλο ευχαστιστώ στην οικογένεια μου που με στήριξε ώστε να καταφέρω να φέρω εις πέρας τις σπουδές μου. Τέλος, να ευχαριστήσω και τους φίλους μου που στάθηκαν δίπλα μου όλο αυτό το διάστημα.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ντζιός Ιωάννης
Βόλος
Σεπτέμβριος 2019

Περίληψη

Πέραν των συνηθισμένων πολιτικών αντικατάστασης για προσωρινές μνήμες (cache) για τα κοινά στοιχεία που αποθηκεύονται εκεί, δεν έχει γίνει ιδιαίτερη μελέτη στο πως θα λειτουργούσαν κάποιες νέες πολιτικές αντικατάστασης για blocks από blockchain. Σκοπός της παρούσας εργασίας είναι να προσομοιώσει αρχικά ένα περιβάλλον το οποίο δέχεται αιτήματα για blocks, καθώς και να αναδείξει μετά από πειράματα ποιά ή ποιές πολιτικές αντικατάστασης cache λειτουργούν καλύτερα σε εφαρμογές που συνδέονται άμεσα με την έννοια του blockchain. Τέλος, η μελέτη γίνεται σε διαχωρισμένη σε τμήματα cache (segmented cache), με σκοπό να αναδείξει την σημαντικότητα του κάθε τμήματος καθώς και λειτουργίες αυτού του τύπου μνήμης.

Λέξεις κλειδιά:
Cache, Segmented Cache, Blockchain

Abstract

In addition to the usual cache replacement policies for shared data stored there, there has been no actual study on how new replacement policies would work for blocks by blockchain. The purpose of this paper is to initially simulate an environment that accepts block requests and to highlight, after experiments, which cache replacement policies work better in applications that are directly related to the concept of blockchain. Finally, the study is done in segmented cache to highlight the importance of each segment as well as functions of this type of cache memory.

Keywords:
Cache, Segmented Cache, Blockchain

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

Με τον όρο caching εννοούμε την προσωρινή αποθήκευση δημοφιλών δεδομένων στην μνήμη, με σκοπό την γρήγορη προσπέλαση τους σε περίπτωση που ζητηθούν ξανά. Το caching παίζει πλέον σημαντικό ρόλο σε πολλές εφαρμογές, αφού η ανάγκη για γρήγορη προσπέλαση δεδομένων ολοένα και αυξάνεται. Στην ουσία οι μνήμες cache είναι πολύ μικρές μνήμες και αυτό κάνει την προσπέλαση τους αρκετά γρήγορη.
Το μέγεθος τους όμως δημιούργησε την ανάγκη ανάπτυξης πολιτικών αντικατάστασης [1] με τις πιο διαδεδομένες να είναι οι LRU , FIFO, LIFO κ.λ.π.
Πιο αναλυτικα, όταν η μνήμη cache γεμίσει, εφαρμόζεται ένας αλγόριθμος ο οποίος αφαιρεί από αυτήν το λιγότερο σημαντικό στοιχείο της. Πρέπει είναι να γίνει κατανοητό πως ανάλογα με την περίπτωση του caching, εφαρμόζεται και η κατάλληλη πολιτική αντικατάστασης. Ιδιαίτερο ενδιαφέρον παρουσιάζει η περίπτωση της προσωρινής αποθήκευσης δεδομένων σε σχέση με το blockchain, και πιο συγκεκριμένα με blocks του blockchain. Η παραπάνω έννοια θα αναλυθεί περαιτέρω σε επόμενο κεφάλαιο.

1.1 Τεχνολογικό υπόβαθρο

Για να είναι σε θέση ο αναγνώστης να κατανοήσει την παρούσα εργασία θα πρέπει να γνωρίζει την έννοια της μνήμης cache , βασικούς αλγόριθμους αντικατάστασης που εφαρμόζονται σε αυτές καθώς και την έννοια του blockchain. Στα κεφάλαια 2 και 3 γίνεται εκτενής περιγραφή της δομής cache που θα μελετηθεί καθώς και της δομής blockchain. Εκτός από τα παραπάνω, για την προσομοίωση των δεδομένων, γίνεται χρήση της κατανομής που υπακούει στον zipf νόμο.

Οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν είναι η Java και ένα framework της Javascript, η NodeJS [2]. Για την αναπαράσταση της μνήμης cache χρησιμοποιήθηκε η δομή LinkedHashMap [3]. Τέλος, για τα γραφήματα που παράχθηκαν κατά την διαδικασία των πειραμάτων, χρησιμοποιήθηκε το εργαλείο plot.ly [4].

1.2 Δομή

Στο κεφάλαιο 2 περιγράφεται η έννοια, οι χρήσεις καθώς και η δομή του blockchain.

Στο κεφάλαιο 3 γίνεται αναφορά στον τύπο μνήμης ςαςηε η οποία χωρίζεται σε τμήματα. Παρουσιάζεται επίσης η δομή της με παραδείγματα.

Στο κεφάλαιο 4 παραθέτονται οι αλγόριθμοι αντικατάστασης που προέκυψαν από την μελέτη της εργασίας καθώς και οι αντίστοιχοι ψευδοκώδικες.

Στο κεφάλαιο 5 παρουσιάζονται τα δεδομένα και ο τρόπος που αυτά παράχθηκαν για τις ανάγκες του πειράματος. Γίνεται αναφορά επίσης στην κατανομή zipf.

Το κεφάλαιο 6 περιγράφει τα πειράματα και παραθέτει αποτελέσματα.

Τέλος, στο κεφάλαιο 7 αναφέρονται τα συμπεράσματα που προέκυψαν από την παρούσα έρευνα καθώς και προτάσεις για μελλοντική εργασία.

1.3 Σκοπός

Η παρούσα εργασία έχει ως σκοπό να αναδείξει νέες πολιτικές αντικατάστασης σε περιβάλλοντα όπου τα δεδομένα που αποθηκεύονται προσωρινά στη μνήμη είναι blocks από blockchain. Αυτές οι πολιτικές αντικατάστασης προκύπτουν από τα πειράματα και την μελέτη των σημαντικών στοιχείων ενός block, όπως για παράδειγμα το μέγεθος του block, η δυσκολια εξόρυξης κ.ο.κ , διαδικασία η οποία περιγράφεται σε επόμενα κεφάλαια. Στόχος επίσης είναι να αναδείξουμε και την απόδοση της μνήμης cache μέσα από τον διαχωρισμό της μνήμης σε τμήματα και την εκτέλεση των πειραμάτων στον συγκεκριμένο τύπο μνήμης.

1.4 Αναγκαιότητα έρευνας

Η δομή του blockchain είναι υλοποιημένη με τέτοιο τρόπο που δεν μπορεί να διασπαστεί. Αυτό σημαίνει πως αν ένα σύστημα θέλει να έχει πρόσβαση σε αυτή, τότε θα πρέπει να αποθηκεύσει στη μνήμη του ολόκληρη τη δομή.

Αυτό μπορεί να αποτελέσει σημαντικό πρόβλημα στη μνήμη των συστημάτων που τη χρησιμοποιούν, μιας και το μέγεθος του blockchain συνεχώς αυξάνεται όσο προστίθενται blocks σε αυτό. Κάθε κόμβος αναγκάζεται να αποθηκεύσει ολόκληρη τη δομή, ενώ πιθανότατα να χρειάζεται μόνο ένα μικρό μέρος αυτής.

Στην παρούσα έρευνα, υλοποιείται ένα μοντέλο μνήμης η οποία αποθηκεύει μόνο τα σημαντικά για τον κάθε κόμβο blocks του blockchain. Η σημαντικότητα του κάθε block για τον κόμβο υπολογίζεται από αλγορίθμους προτίμησης που περιγράφονται σε επόμενο κεφάλαιο.

1.5 Περιορισμοί της εργασίας

Η παρούσα εργασία έχει σκοπό να προσομοιώσει την αποθήκευση block από blockchain σε μια μνήμη cache. Γι'αυτό το λόγο δόθηκε βάρος στη δομή της μνήμης καθώς και στην δημιουργία των πολιτικών αντικατάστασης στη μνήμη και όχι τόσο στα πραγματικά δεδομένα.

Τα blocks για την εκτέλεση των πειραμάτων δεν είναι πραγματικά blocks, αλλά μια προσομοίωση σε μορφή JSON [5].

Αρχικά για το αναγνωριστικό κάθε block (block id) για το σύστημα που υλοποιήθηκε είναι ένας ακέραιος αριθμός και όχι η πραγματική hash τιμή του. Ομοίως και ο δείκτης του block που δείχνει το προηγούμενο block στην αλυσίδα θεωρείται ως ακέραιος. Επίσης οι τιμές για την δυσκολία εξόρυξης καθώς και το μέγεθος του block έχουν προσομοιωθεί ώστε να συμβαδίζουν όσο το δυνατόν περισσότερο με τις πραγματικές τιμές.

Τέλος, δεν έχουν ληφθεί υπόψη οι πραγματικές συναλλαγές που περιλαμβάνει κάθε block, μιας και μελετάται μονο το μέγεθος τους. (transactions count)

1.6 Κώδικας Εργασίας

Ο κώδικας της παρούσας εργασίας βρίσκεται στο Github και είναι διαθέσιμος στο παρακάτω link:
`https://github.com/ntzios10/blockchainCache`

Η εφαρμογή η οποία προσομοιάζει την cache είναι γραμμένη σε Java.

Κεφάλαιο 2

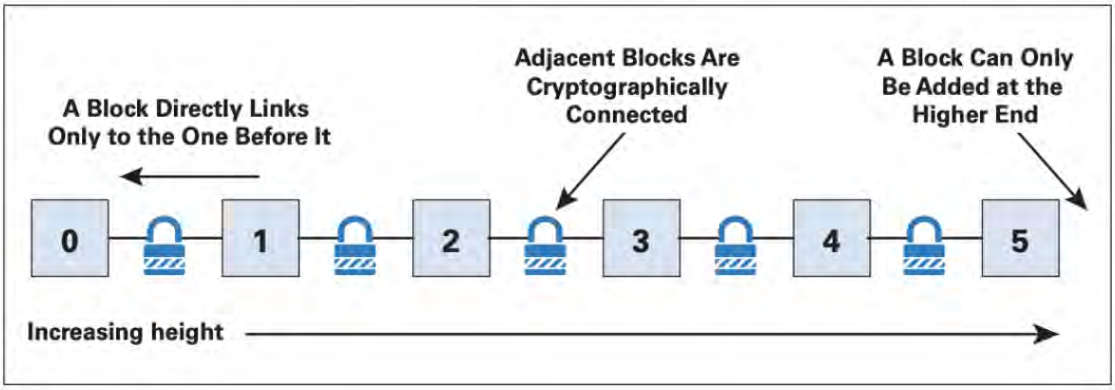
Blockchain

Σε αυτήν την ενότητα περιγράφεται και επεξηγείται η έννοια του Blockchain, το από τι αποτελείται ένα blockchain καθώς και οι τομείς στους οποίους βρίσκει εφαρμογή.

2.1 Τι είναι το Blockchain

Το Blockchain [6] είναι μια κατανεμημένη δομή δεδομένων που περιλαμβάνει εγγραφές οι οποίες είναι μεταξύ τους συνδεδεμένες σαν αλυσίδα. Στην ουσία, αυτή η τεχνολογία δίνει τη δυνατότητα για τη δημιουργία μιας αλυσίδας δεδομένων, η οποία μοιράζεται σε ένα κατανεμημένο δίκτυο υπολογιστών. Πιο συγκεκριμένα, κάθε block περιλαμβάνει ένα κρυπτογραφημένο σύνδεσμο προς το προηγούμενο block της αλυσίδας, την χρονική στιγμή της δημιουργίας του καθώς και κάποια metadata που περιέχουν συναλλαγές και άλλες χρήσιμες πληροφορίες. Κάθε φορά που ένα block πρέπει να προστεθεί στην αλυσίδα, προστίθεται στο τέλος αυτής με ένα link προς το τελευταίο block.

Η κρυπτογράφηση του συνδέσμου επιτυγχάνεται μέσω του αλγορίθμου SHA-256 [7]. Με τον τρόπο που έχει σχεδιαστεί λοιπόν η συγκεκριμένη δομή, εγγυάται την ασφάλεια και την ακεραιότητα των δεδομένων.



Σχήμα 2.1: Αναπαράσταση σύνδεσης των blocks της αλυσίδας [6].

2.2 Χρήσεις

Τα κρυπτονομίσματα είναι η πιο διαδεδομένη εφαρμογή του blockchain. Το πιο δημοφιλές είναι το bitcoin το οποίο έχει παρουσιάσει τεράστια ανάπτυξη τα τελευταία χρόνια, με την τιμή του τον Δεκέμβρη του 2017 να φτάνει τα 19,783.06 δολάρια [8].

Εκτός από τα κρυπτονομίσματα όμως, η τεχνολογία blockchain βρίσκει κι' άλλες εφαρμογές σε διάφορους τομείς. Ενδεικτικά μερικές από αυτές είναι οι εξής:

- Επεξεργασία ασφαλιστικών αιτήσεων.
- Ιατρικές καρτέλες μεταξύ ασθενών και παρόχων.
- Δεδομένα για πληθυσμούς ανθρώπων.
- Εφαρμογές για την διαχείριση ψηφοφοριών.
- Παρακολούθηση θεμάτων υγείας ασθενών και αυτοματοποίηση σχετικών διαδικασιών.
- Εκτέλεση μικροπληρωμών (για παράδειγμα pay-as-you-go ασφάλιση).
- Αυτοματοποίηση σε πληρωμές μέσω κουπονιών.
- Διαχείριση συστήματος πληρωμών σε σταθμούς φόρτισης ηλεκτρικών οχημάτων.
- Καταγραφή πληροφοριών και ιστορικού προϊόντων.

2.3 Block

2.3.1 Δομή Block

Η δομή ενός block αποτυπώνεται στον παρακάτω πίνακα.

Item
Magic Number
Block size
Version Number
Backward Link
Transaction Hash
Timestamp
Mining Difficulty
Nonce
Transaction Count
Transaction 1
Transaction 2
• • •
Transaction N

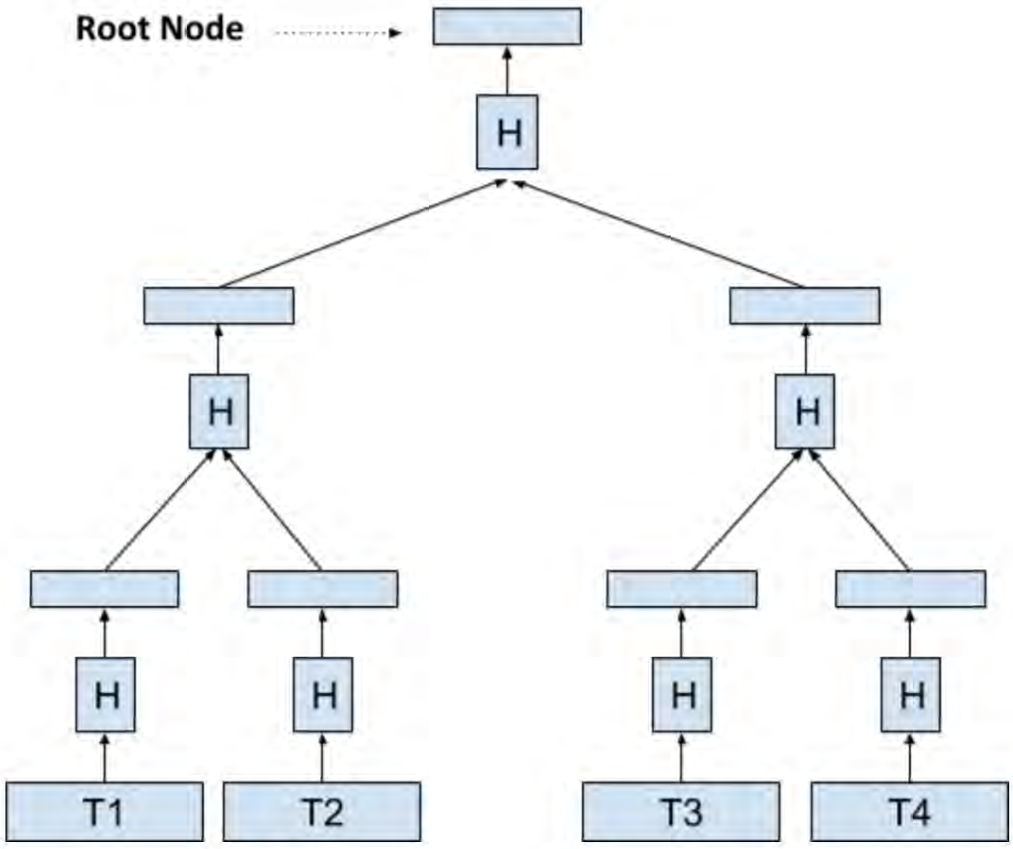
Σχήμα 2.2: Περιεχόμενα ενός block [6].

Παρακάτω, αναλύεται το κάθε πεδίο του block:

- **Magic Number**: Πρόκειται για το μοναδικό αναγνωριστικό (unique id) του κάθε block ανάμεσα στα υπόλοιπα της αλυσίδας. Δεν μπορεί να αλλάξει ποτέ και κάθε block έχει διαφορετικό id από τα υπόλοιπα.
- **Blocksize**: Το νούμερο των bytes του block.
- **Version Number**:Αυτό το πεδίο υποδεικνύει την έκδοση του block. Για παράδειγμα, σε περίπτωση που η δομή του block χρειαστεί να αλλάξει, το version number θα αυξηθεί. Ο λόγος που αυτό το πεδίο είναι αναγκαίο είναι γιατί το blockchain θα πρέπει να είναι backward compatible με τις παλαιότερες εκδόσεις του.
- **Backward Link**: Το Backward Link του κάθε block παράγεται από όλα τα header πεδία του προηγούμενου block στην αλυσίδα. Τα header πεδία ενός item είναι τα παρακάτω:
  - Version Number
  - Backward Link
  - Transaction Hash
  - Timestamp
  - Mining difficulty
  - Nonce

  Στην δημιουργία του, τα παραπάνω πεδία κρυπτογραφούνται με τον αλγόριθμο SHA-256 και παράγεται το backward link. Η συγκεκριμένη διαδικασία είναι αυτή που πιστοποιεί την ασφάλεια και την μη αναστρεψιμότητα σε περίπτωση που κάποιος προσπαθήσει να αλλάξει ένα block.
- **Timestamp**: Η χρονική στιγμή στην οποία το block δημιουργήθηκε.
- **Mining Difficulty**: Πρόκειται για το πεδίο που περιγράφει το πόσο δύσκολο είναι να παραχθεί και να προστεθεί στην αλυσίδα ένα νέο block.
- **Nonce**: Είναι ένα τυχαίο / ψευδο-τυχαίο νούμερο το οποίο χρησιμοποιείται μια φορά σε κρυπτογραφημένες επικοινωνίες. Χρησιμοποιείται στον υπολογισμό του proof-of-work.
- **Transaction Count**: Ο αριθμός των συναλλαγών που περιλαμβάνει το block στα data του.
- **Transactions**: Η λίστα των συναλλαγών του block. Το μέγεθος της παραπάνω λίστας είναι ίσο με το Transaction Count που περιγράφεται πάνω.

- **Transaction Hash**: Πρόκειται για τη ρίζα του Merkle tree. Πιο συγκεκριμένα, όλες οι συναλλαγές που περιλαμβάνει ένα block βρίσκονται στα φύλλα αυτού του δέντρου. Αυτές κρυπτογραφούνται με τον αλγόριθμο SHA-256 έτσι ώστε να παραχθεί τελικά η ρίζα του δέντρου, η οποία είναι απόγονος όλων των κρυπτογραφημένων συναλλαγών. Σε παρακάτω εικόνα αποτυπώνεται το Merkle tree.



Σχήμα 2.3: Παράδειγμα του Merkle Tree [6]

2.3.2 Αξιοποίηση πεδίων

Στην παρούσα εργασία έχουν επιλεχθεί να αξιοποιηθούν τα παρακάτω πεδία:

- **Block size**: Το μέγεθος του block είναι μια πολύ σημαντική ποσότητα για την διαδικασία των πειραμάτων αφού δείχνει το πόσο μεγάλο είναι, συνεπώς το πόση πληροφορία περιλαμβάνει.
- **Mining Difficulty**: Η δυσκολία δημιουργίας ενός block από τους miners είναι εξίσου σημαντικός παράγοντας που μας δείχνει εάν ένα block είναι σημαντικό στο σύνολο του blockchain. Αυτή η πληροφορία λαμβάνεται υπόψη στα πειράματα την μνήμη cache που αναλύονται σε επόμενο κεφάλαιο.
- **Transaction Counter**: Ένας ακόμη παράγοντας που λαμβάνεται υπόψη είναι το σύνολο των συναλλαγών που περιλαμβάνει ένα block. Όσες περισσότερες συναλλαγές περιλαμβάνει, τόσο πιο σημαντικό είναι.

Κεφάλαιο 3

Cache

Στο κεφάλαιο 3 περιγράφεται η δομή της μνήμης καθώς επεξηγούνται και τα μέρη αυτής αναλυτικά.

3.1 Δομή μνήμης

Η προσωρινή μνήμη (cache) που έχει χρησιμοποιηθεί για την παρούσα έρευνα χωρίζεται σε τμήματα. Πρόκειται για ένα διαχωρισμό της μνήμης ούτως ώστε το κάθε τμήμα της να κρατάει διαφορετικής δημοφιλίας δεδομένα. Ο διαχωρισμός της σε τμήματα δεν είναι τυχαίος, μιας και ορίζουμε ένα αριθμό λ ο οποίος δείχνει το μέγεθος του ενός τμήματος σε σχέση με το συνολικό μέγεθος. Για την μελέτη των πειραμάτων η cache έχει χωριστεί σε 2 τμήματα, το τμήμα προθέρμανσης και το κύριο τμήμα, τα οποία θα αναλυθούν περεταίρω παρακάτω.

Η σταθερά λ υπολογίζεται από την παρακάτω εξίσωση:

$$\lambda = \frac{earlypartSize}{totalcacheSize} \tag{3.1}$$

Οι μεταβλητή totalcacheSize δηλώνει το συνολικό μέγεθος της μνήμης cache ενώ η μεταβλητή earlypartSize δηλώνει το μέγεθος του τμήματος προθέρμανσης. Η μεταβλητή λ κατά την πειραματική διαδικασία παίρνει τιμές από **0.5** ως **0.8**. Πιο συγκεκριμένα , πρόκειται για το ποσοστό επί τοις εκατό του μεγέθους του τμήματος προθέρμανσης, ως προς το συνολικό μέγεθος της μνήμης.

3.2 Τμήματα μνήμης

3.2.1 Τμήμα προθέρμανσης

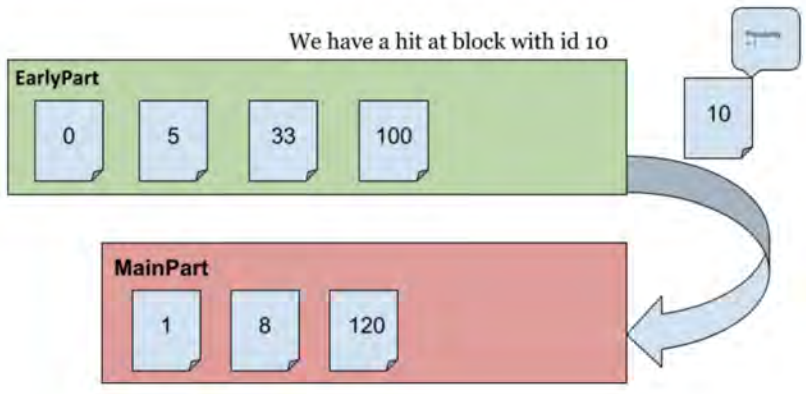
Πρόκειται για το μεγαλύτερο σε μέγεθος τμήμα της μνήμης cache. Εδώ αποθηκεύονται τα λιγότερο δημοφιλή στοιχεία, τα οποία με το που αποκτήσουν την απαιτούμενη δημοφιλία, μεταφέρονται στο κύριο τμήμα. Για να προσδιοριστεί το πότε ένα στοιχείο της cache είναι αρκετά δημοφιλές, ορίζουμε ένα popularity threshold το οποίο ανάλογα με το συνολικό μέγεθος της cache μπορεί να πάρει τιμές από 1 μέχρι 5. Όταν ένα στοιχείο εισέρχεται στο τμήμα προθέρμανσης για πρώτη φορά τότε η δημοφιλία του ισούται με 1. Στην περίπτωση που το συγκεκριμένο στοιχείο ξανα ζητηθεί, η δημοφιλία του αυξάνεται κατά μία μονάδα. Τελικά, όταν η δημοφιλία του ξεπεράσει το threshold που έχουμε ορίσει για το συγκεκριμένο τμήμα της cache, αυτό μεταφέρεται στο τμήμα που προορίζεται για να αποθηκεύει τα πιο δημοφιλη δεδομένα.

3.2.2 Κύριο τμήμα

Το κύριο τμήμα της μνήμης είναι αυτό στο οποίο αποθηκεύονται τα πιο δημοφιλή blocks. Λόγω αυτού του διαχωρισμού, οι πιο “hot” εγγραφές συγκεντρώνονται σε ένα κομμάτι της cache, ούτως ώστε να είναι αρκετά πιο εύκολη η προσπέλαση τους.

Και στα δύο τμήματα (προθέρμανσης και κύριο) εφαρμόζονται πολιτικές αντικατάστασης στην περίπτωση που γεμίσουν από blocks. Οι πολιτικές αυτές επεξηγούνται λεπτομερώς σε επόμενα κεφάλαια.

Στην παρακάτω εικόνα φαίνεται η δομή της μνήμης η οποία είναι χωρισμένη στα 2 τμήματα που αναφέρθηκαν πιο πάνω. Όταν ένα block του τμήματος προθέρμανσης γίνει αρκετά δημοφιλές, τότε μεταφέρεται στο κύριο τμήμα της μνήμης. Στο παράδειγμα της εικόνας, το block με id 10, έχει δημοφιλία μεγαλύτερη της μονάδας, και για αυτό το λόγο μεταφέρεται στο κατάλληλο τμήμα της μνήμης.



Σχήμα 3.1: Η μνήμη cache χωρισμένη σε τμήματα.

Κεφάλαιο 4

Αλγόριθμοι

Όταν η μνήμη cache γεμίζει πρέπει να εφαρμοστεί κάποια πολιτική αντικατάστασης ώστε να φύγει η πιο ασήμαντη εγγραφή και να αποθηκευτεί μια νέα. Στα πλαίσια της παρούσας εργασίας , εκτός των κοινών πολιτικών αντικατάστασης, έχει δοθεί έμφαση στην σημαντικότητα του block ανάλογα με τις τιμές των πεδίων του. Στις επόμενες ενότητες αυτού του κεφαλαίου αναλύονται οι αλγόριθμοι πάνω στους οποίους θα εκτελεστούν τα πειράματα.

4.1 Σημαντικότητα block

Στην προσομοίωση της cache, όταν ένα block πρόκειται να προστεθεί στη μνήμη, υπολογίζεται για αυτό ένα νέο πεδίο, αυτό της σημαντικότητας του. Το πόσο σημαντικό είναι ένα block καθορίζεται από τα πεδία του. Στο κεφάλαιο 2 έχουν αναφερθεί τα πεδία που περιλαμβάνει κάθε block, καθώς και το τι σημαίνει το καθένα. Με βάση αυτά, προέκυψαν οι παρακάτω αλγόριθμοι όσον αφορά την σημαντικότητα του.

4.1.1 Mining Difficulty Significance (MDS)

Ένας σημαντικός παράγοντας ο οποίος μπορεί να καθορίσει το πόσο σημαντικό είναι ένα block είναι η δυσκολία εξόρυξης του (Mining Difficulty). Όσο πιο δύσκολο είναι για τους miners να παραγάγουν ένα block, τόσο πιο σημαντικό το κάνει.

Ο μαθηματικός τύπος για τον υπολογισμό της σημαντικότητας του block με βάση την δυσκολία εξόρυξης του αποτυπώνεται παρακάτω:

$$BSF = \frac{BSize + 2 * MD + TC}{BSize + MD + TC} \tag{4.1}$$

4.1.2 Transactions Counter Significance (TCS)

Ο αριθμός των συναλλαγών που περιλαμβάνει κάθε block είναι επίσης ένας παράγοντας που καθορίζει το πόσο σημαντικό είναι. Όταν ένα block περιλαμβάνει αρκετές συναλλαγές τότε σημαίνει ότι είναι χρήσιμο και σημαντικό. Στην συγκεκριμένη περίπτωση στο block προστίθεται το πεδίο significance το οποίο δίνεται από τον παρακάτω τύπο:

$$BSF = \frac{BSize + MD + 2 * TC}{BSize + MD + TC} \tag{4.2}$$

4.1.3 Block Size Significance (BSS)

Ένα ακόμη πεδίο του block που υποδεικνύει το εάν αυτό είναι σημαντικό ή όχι είναι το μέγεθος του. Ένα μεγάλο block θα μπορούσε να θεωρηθεί πως είναι επίσης σημαντικό μιας και το μέγεθος του δηλώνει πως το block περιέχει αρκετή πληροφορία. Για τον καθορισμό του significance του block εφαρμόζεται ο παρακάτω τύπος:

$$BSF = \frac{2 * BSize + MD + TC}{BSize + MD + TC} \tag{4.3}$$

Επεξήγηση μεταβλητών:

BSF Block significance

BSize Block size

MD Mining difficulty

TC Transaction counter

Οι παραπάνω μεταβλητές έχουν υποστεί κανονικοποίηση ως προς τι μέγιστες τιμές τους. Οι μέγιστες τιμές για την κάθε μεταβλητή είναι οι εξής:

$\mathbf{BSize}_B$: 1500

$\mathbf{MD}_B$: 100

$\mathbf{TC}_B$: 200

4.2 Δημοφιλία block

Κατά την εισαγωγή ενός block στην μνήμη προστίθεται ένα ακόμη πεδίο, αυτό της δημοφιλίας του (popularity). Όταν ένα block μπαίνει για πρώτη φορά, τότε το πεδίο που δηλώνει την δημοφιλία του αρχικοποιείται στην τιμή 1. Όταν ζητηθεί για δεύτερη φορά, τότε ανανεώνεται η τιμή αυτού του πεδίου καθώς αυξάνεται κατά μια μονάδα. Με βάση αυτό το πεδίο, η cache μπορεί να υποστηρίξει μια πολιτική αντικατάστασης κατά την οποία, το πιο ασήμαντο στοιχείο της, είναι αυτό που έχει την χαμηλότερη δημοφιλία.

4.3 Σειρά εισαγωγής block στην cache (Insertion Order Significance)

Ένα ακόμη στοιχείο όπου δηλώνει εάν ένα block πρέπει να κρατηθεί στη μνήμη διότι ίσως χρειαστεί ξανά στο μέλλον είναι το πότε αυτό εισήχθει στην cache για πρώτη φορά. Όταν εισάγεται ένα block στην cache κρατάμε έναν μετρητή ο οποίος δείχνει το ποιά εισαγωγή είναι αυτή που εκτελείται εκείνη τη δεδομένη στιγμή. Η μεταβλητή αυτή αρχικοποιείται στο 0 πριν την εκτέλεση του πειράματος, και αυξάνεται όσο εκτελείται το πείραμα. Για παράδειγμα το 1ο block που προστίθεται στην cache έχει insertion order ίσο με τη μονάδα, το 2ο δύο κ.ο.κ. Με αυτόν τον τρόπο το block με τη μεγαλυτερη σειρά εισαγωγής θεωρείται πιο σημαντικό από ένα άλλο που έχει μικρότερη τιμή, δηλαδή που έχει εισαχθεί πιο παλιά στη μνήμη.

4.4 Ψευδοκώδικες

Σε αυτή την ενότητα περιγράφονται οι ψευδοκώδικες των αλγορίθμων που αναλύθηκαν σε αυτό το κεφάλαιο.

Algorithm 1 Mining Difficulty Significance

```
1: for each: block_i ∈ inputDataset do
2:     significancePolicy ← MDS
3:     if block_i ∈ cache then
4:         updateBlockInCache();
5:         hits++
6:         exit;
7:     else
8:         addBlockInCache();
9:     end if
10: end for
11: hitRatio ← totalHits/blocks.length
12: print(hitRatio)
```

Algorithm 2 Transaction Counter Significance

```
1: for each: block_i ∈ inputDataset do
2:     significancePolicy ← TCS
3:     if block_i ∈ cache then
4:         updateBlockInCache();
5:         hits++
6:         exit;
7:     else
8:         addBlockInCache();
9:     end if
10: end for
11: hitRatio ← totalHits/blocks.length
12: print(hitRatio)
```

Algorithm 3 Block Size Significance

```
1: for each: block_i ∈ inputDataset do
2:     significancePolicy ← BSS
3:     if block_i ∈ cache then
4:         updateBlockInCache();
5:         hits++
6:         exit;
7:     else
8:         addBlockInCache();
9:     end if
10: end for
11: hitRatio ← totalHits/blocks.length
12: print(hitRatio)
```

Algorithm 4 Insertion Order Significance

```
1: for each: block_i ∈ inputDataset do
2:     significancePolicy ← IOS
3:     if block_i ∈ cache then
4:         updateBlockInCache();
5:         hits++
6:         exit;
7:     else
8:         addBlockInCache();
9:     end if
10: end for
11: hitRatio ← totalHits/blocks.length
12: print(hitRatio)
```

Οι ψευδοκώδικες των συναρτήσεων updateBlockInCache() και addBlockInCache():

Algorithm 5 Update Block in Cache

```
1: blockPopularity + +
2: if blockPopularity > 1 then
3:     if mainPart has space then
4:         Insert block at mainPart
5:     else
6:         Remove lowest significant block
7:         Insert block at main part
8:     end if
9: else
10:    Add block at early part
11: end if
```

Algorithm 6 Add Block in Cache

```
1: if earlyPart has space then
2:     Insert block at earlyPart
3: else
4:     Remove lowest significant block
5:     Insert block at early part
6: end if
7: totalInsertions++
```

Για τον αλγόριθμο που αφαιρεί το λιγότερο δημοφιλές block από τη μνήμη γίνεται μια διαφοροποίηση στις παραπάνω συναρτήσεις (addBlockInCache, updateBlockInCache) και στα βήματα όπου αφαιρείται το λιγότερο σημαντικό block, καλείται η συνάρτηση η οποία αφαιρεί το λιγότερο δημοφιλές block από τη μνήμη.

Algorithm 7 Remove Lowest Significant Block

```
1: lowestSignificantBlock ← null
2: lowestSignificance ← −∞
3: for each: block_i ∈ cachePart do
4:     if block_i significance > lowestSignificance then
5:         lowestSignificantBlock ← block_i
6:         lowestSignificance ← block_i significance
7:     end if
8: end for
9: return lowestSignificantBlock
```

Algorithm 8 Remove Lowest Popular Block

```
1: lowestPopularBlock ← null
2: lowestPopularity ← −∞
3: for each: block_i ∈ cachePart do
4:     if block_i popularity > lowestPopularity then
5:         lowestPopularBlock ← block_i
6:         lowestPopularity ← block_i popularity
7:     end if
8: end for
9: return lowestPopulartBlock
```

Κεφάλαιο 5

Δεδομένα πειράματος

Τα δεδομένα των πειραμάτων έχουν προσομοιωθεί και έχουν παραχθεί από εφαρμογή που δημιουργήθηκε για τον σκοπό αυτό. Πρόκειται για ένα σύνολο από blocks σε JSON [5] μορφή τα οποία περιλαμβάνουν τυχαίες τιμές για την εκτέλεση των πειραμάτων. Ο τρόπος που παράγονται δεν είναι τυχαίος, καθώς μια τέτοια κατανομή από blocks δεν αντιπροσωπεύει την πραγματικότητα. Είναι λογικό πως κάποια blocks θα εμφανίζονται περισσότερες φορές από κάποια άλλα. Οι κατανομές που χρησιμοποιήθηκαν για την δημιουργία του data set καθώς και ο τρόπος που προσομοιώθηκαν περιγράφονται σε αυτό το κεφάλαιο.

5.1 Zipfian κατανομή

Αυτή η κατανομή αναφέρεται σε φαινόμενα όπου έχουμε πολλές εμφανίσεις μικρών γεγονότων και λίγες εμφανίσεις μεγάλων γεγονότων. Για παράδειγμα, έχουμε πολλές μικρές πόλεις ενώ παράλληλα έχουμε λίγες μεγάλες. Ένα ακόμη παράδειγμα είναι και η συχνότητα χρήσης λέξεων μιας γλώσσας, μιας και υπάρχουν λέξεις οι οποίες χρησιμοποιούνται αρκετά συχνά, ενώ άλλες αρκετά σπάνια. Ο δυναμο-νόμος [9] περιγράφεται από την παρακάτω εξίσωση:

$$r * f = k \tag{5.1}$$

Όπου r είναι είναι η σειρά εμφάνισης ενός γεγονότος και f είναι η συχνότητα εμφάνισης του. Το γινόμενο αυτό είναι σχεδόν σταθερό ίσο με k. Στο παρακάτω γράφημα απεικονίζεται γραφικά το πως κατανέμονται τα δεδομένα σε μια κατανομή που ακολουθεί τον νόμο του zipf.

Σχήμα 5.1: Κατανομή νόμου δύναμης [9].

Στην παραπάνω εικόνα παρουσιάζεται ακριβώς το πως κατανέμονται τα δεδομένα σε μια zipf κατανομή. Στα αριστερά φαίνονται οι λίγες τιμές που εμφανίζονται συχνά, ενώ δεξιά, στην ουρά της γραφικής παράστασης εμφανίζονται οι πιο σπάνιες εμφανίσεις δεδομένων.

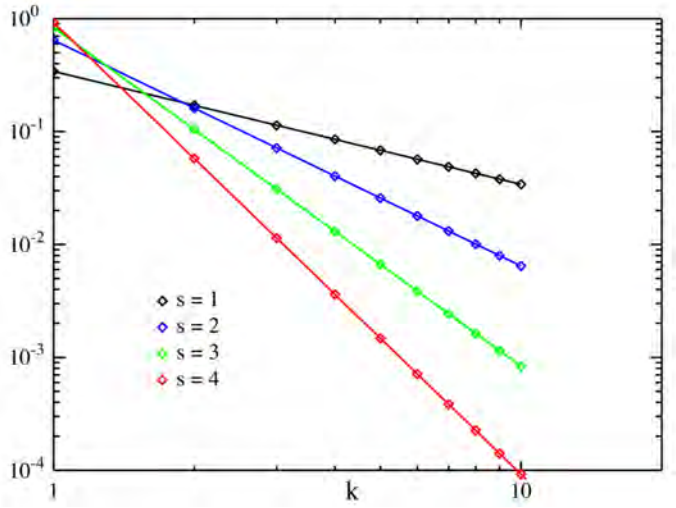
5.2 Παραλλαγές εκθέτη zipf

Αναλύοντας τον νόμο του zipf [10][11] λίγο περισσότερο με μια πιο μαθηματική ερμηνεία θα δούμε πως η συχνότητα εμφάνισης ενός γεγονότος δίνεται από τον παρακάτω τύπο:

$$f(k, s, N) = \frac{(1/k^s)}{\sum_{n=1}^{N} 1/n^s} \tag{5.2}$$

Όταν ο νόμος του zipf εφαρμόζεται για παράδειγμα σε πληθυσμούς πόλεων, τοτε έχει βρεθεί πως η ιδανική τιμή για τον εκθέτη είναι για $s = 1.07$. Η προκαθορισμένη τιμή για τον εκθέτη είναι η μονάδα, όμως για τα πειράματα έχουν εξεταστεί διάφορες τιμές σε εύρος από 0.8 εώς και 2.0 .

Για την πλειοψηφία των πειραμάτων ο εκθέτης πήρε την τιμή 1.1 , τιμή πολύ κοντινή στην προκαθορισμένη τιμή του εκθέτη. Εκτελέστηκαν όμως και πειράματα για μικρότερους καθώς και μεγαλύτερους εκθέτες τα οποία παρουσιάζονται στο επόμενο κεφάλαιο.



Σχήμα 5.2: Zipf PMF (Probability mass function)

5.3 Προσομοίωση

Για την εκτέλεση των πειραμάτων αναπτύχθηκε ένα βοηθητικό λογισμικό που παράγει το σύνολο των δεδομένων που θα δοθούν σαν είσοδος στην μνήμη cache. Πιο συγκεκριμένα, πρόκειται για έναν μηχανισμό που παράγει ένα σύνολο από blocks σε JSON μορφή, με τρόπο τέτοιο ώστε να υπάρχουν αφενός μεν αρκετές ίδιες εγγραφές, αφετέρου δε να παράγονται με τέτοιο τρόπο ώστε να ακολουθούν μια κατανομή. Έπειτα από έρευνα, επιλέχθηκε η κατανομή που ακολουθεί τον νόμο του zipf με τιμή εκθέτη s=1.1 (Ενότητα 5.2). Η διαδικασία παραγωγής των block γίνεται ως εξής.

Δημιουργείται μια zipfian κατανομή ακεραίων μεγέθους N. Παράγονται N blocks όπου σε κάθε block id αντιστοιχίζεται ένας τυχαίος ακέραιος της παραπάνω κατανομής. Έπειτα, για τον καθορισμό του συνδέσμου του block με το προηγούμενο ορίζεται σαν block id το id του N-1 block. Στην συνέχεια ορίζονται με τυχαίο τρόπο τα εξής στοιχεία του κάθε block:

- **Block size** : [900 , 1500 KB]
- **Transactions counter** : [800 , 1200]
- **Mining Difficulty** : [1 , 100]
- **Timestamp**

Το timestamp κάθε block i είναι πάντα μεγαλύτερο από το timestamp του i-1 block. Σημαντική σημείωση είναι ό,τι όταν το τυχαίο block id στην παραγωγή των δεδομένων υπάρχει ήδη στο data set, τότε τα στοιχεία block size , transactions counter , mining difficulty και timestamp παραμένουν ίδια. Αυτό γίνεται ώστε να έχουμε πολλά ίδια blocks ώστε να εφαρμοστεί η πολιτική αντικατάστασης στην μνήμη cache.

Παρακάτω βρίσκεται το git repository του μηχανισμού που περιγράφεται:

`https://github.com/ntzios10/blockArray-generator`

Κεφάλαιο 6

Πειράματα και αποτελέσματα

Στην ενότητα αυτή αναλύονται τα πειράματα που εκτελέστηκαν για τις ανάγκες της παρούσας εργασίας καθώς και αποτυπώνονται γραφικά τα αποτελέσματα των πειραμάτων.

6.1 Πειραματική διαδικασία

Για την υλοποίηση των πειραμάτων παράχθηκε ένα σύνολο από 1000 blocks με τη βοήθεια του μηχανισμού που αναλύθηκε στο κεφάλαιο 5. Πρόκειται για ένα σύνολο που υπακούει στην zipfian κατανομή ούτως ώστε τα δεδομένα και η ζήτηση τους να συμβαδίζουν με την πραγματικότητα. Το μέγεθος της συνολικής μνήμης κυμαίνεται από 20 ως 200 slots, δηλαδή πρόκειται για ένα ποσοστό από 0,02 % ως 0,2 % επί του συνόλου του input data-set.

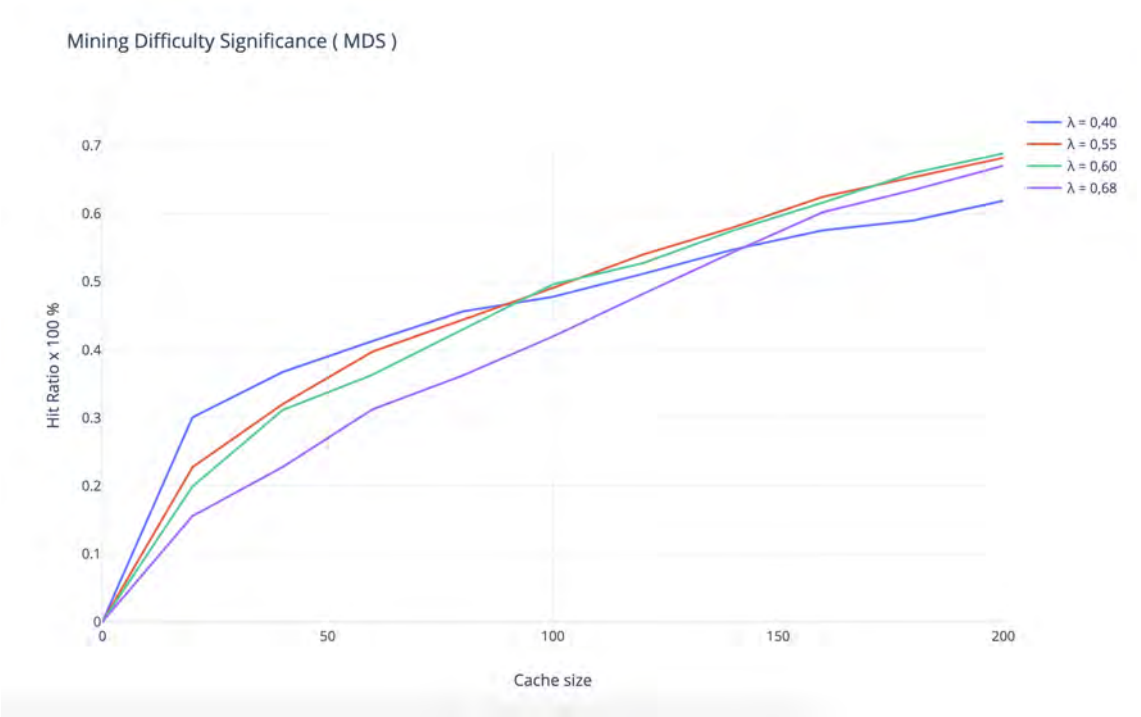
Στην αρχή της πειραματικής διαδικασίας προσδιορίζεται το συνολικό μέγεθος της μνήμης cache καθώς και ο παράγοντας λ που προσδιορίζει το πως το συνολικό μέγεθος κατανέμεται στα μέρη της cache, το τμήμα προθέρμανσης και το κύριο τμήμα. Στην περίπτωση που το μέγεθος της cache είναι 100 θέσεις, τότε για λ = 0,6 το τμήμα προθέρμανσης αρχικοποιείται με 60 θέσεις ενώ το κύριο τμήμα με τις υπόλοιπες 40.

Στα πειράματα, για τις διάφορες τιμές της μεταβλητής λ , ελέγχεται η απόδοση των αλγορίθμων που υλοποιήθηκαν όσον αφορά το hit ratio της μνήμης.

6.2 Αποτελέσματα

6.2.1 Απόδοση για τις τιμές του λ

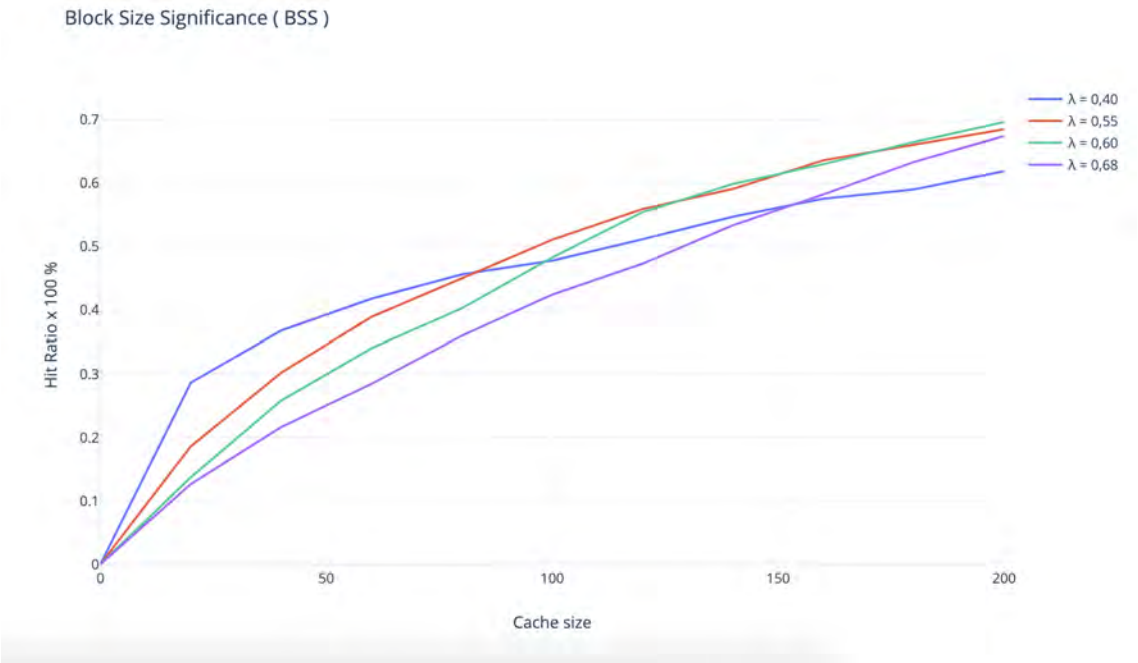
Σε αυτήν την υποενότητα καταγράφεται η απόδοση του κάθε αλγορίθμου συγκριτικά με το μέγεθος των τμημάτων της μνήμης.



Σχήμα 6.1: Απόδοση MDS για διάφορα λ .

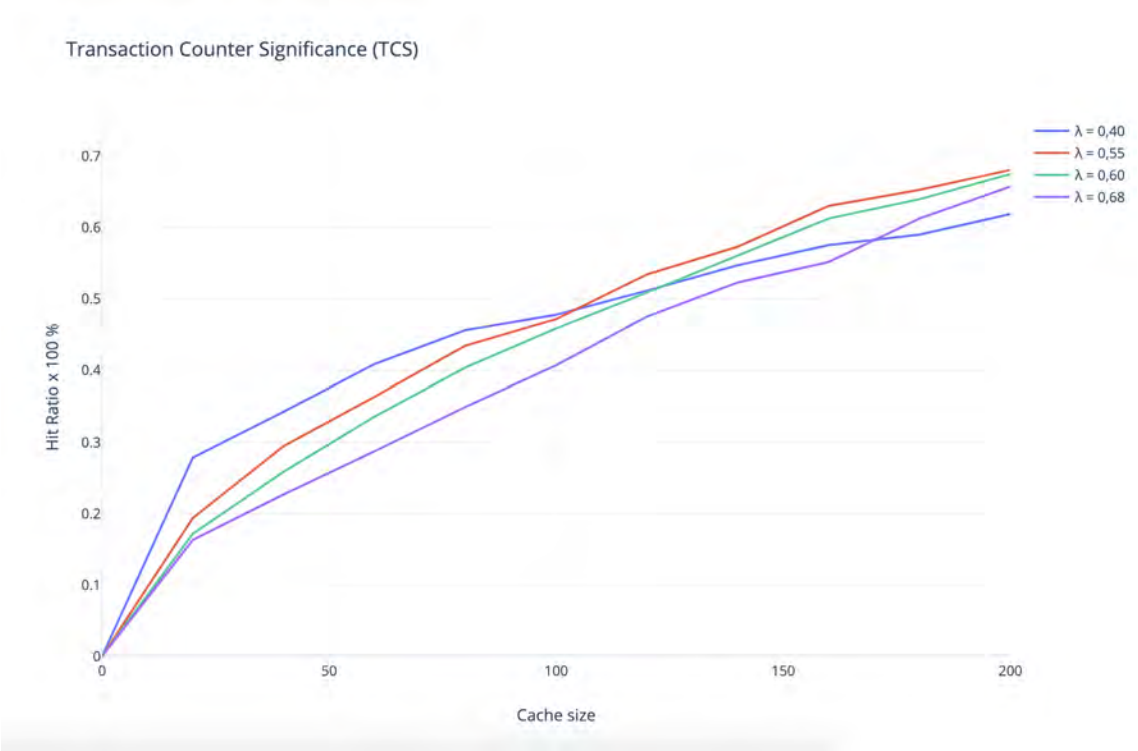
Στο παραπάνω διάγραμμα απεικονίζεται η απόδοση του αλγορίθμου Mining Difficulty Significance για τις διάφορες τιμές της μεταβλητής λ. Παρατηρείται πως όταν το μέγεθος του early part είναι μικρότερο από το main part της μνήμης cache (λ μικρότερο από 0,50) και παράλληλα το συνολικό μέγεθος της μνήμης είναι μικρό, τότε ο αλγόριθμος εμφανίζει περισσότερα hits.

Για το συγκεκριμένο πείραμα το μέγιστο hit ratio εμφανίζεται για λ = 0.60 με την τιμή του να φτάνει στο 0.7019 % ενώ η χειρότερη απόδοση καταγράφεται όταν ο παράγοντας λ πάρει την τιμή 0,40. Σε αυτήν την περίπτωση το ποσοστό των hits επί του συνόλου ανέρχεται στο 0.618 % .



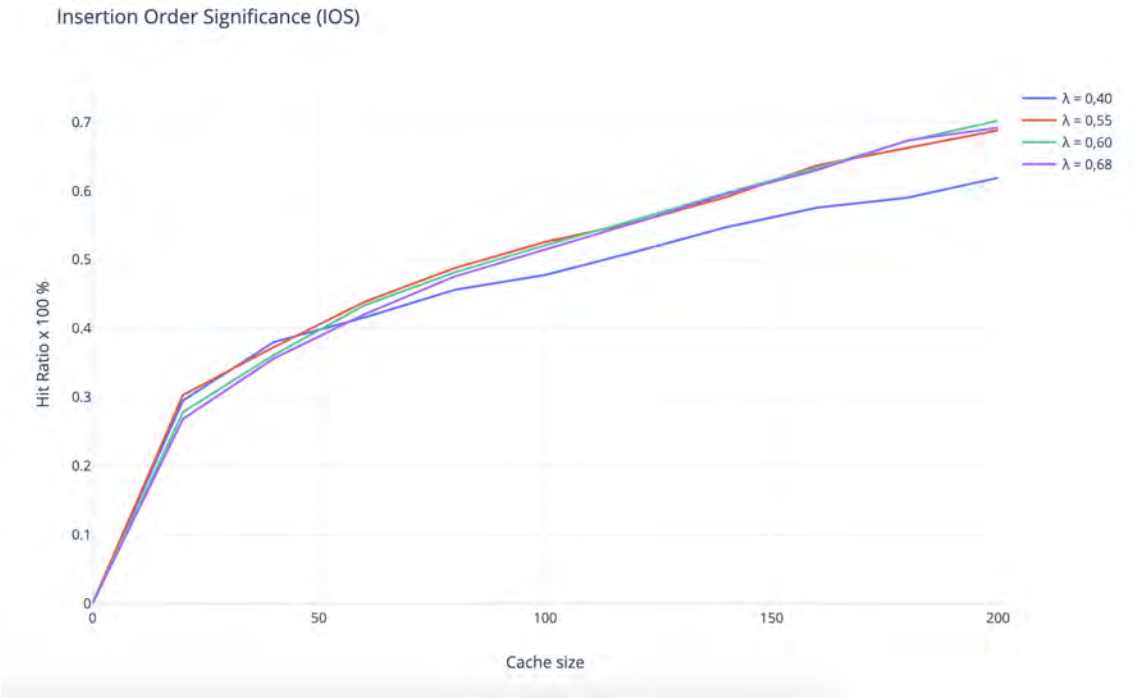
Σχήμα 6.2: Απόδοση BSS για διάφορα λ .

Ομοίως στο παραπάνω διάγραμμα φαίνεται η απόδοση του αλγορίθμου Block Size Significance για τα διάφορα λ που μελετήθηκαν. Η συμπεριφορά αυτού του αλγορίθμου είναι σχεδόν ίδια με αυτή του MD. Χαρακτηριστικά, το μέγιστο hit ratio επιτυγχάνεται για λ = 0,60 . Για αυτήν την τιμή του λ το hit ratio max ισούται με 0.6963 % για την περίπτωση που το συνολικό μέγεθος της μνήμης είναι 200 θέσεις.



Σχήμα 6.3: Απόδοση TCS για διάφορα λ .

Για τον αλγόριθμο Transaction Counter Significance τα αποτελέσματα ακολουθούν το ίδιο μοτίβο, με τη διαφορά οτι το μέγιστο hit ratio επιτυγχάνεται για λ = 0,55 , με την τιμή του να φτάνει στο 0.6807 % για την περίπτωση που η συνολική μνήμη είναι 200 θέσεων.



Σχήμα 6.4: Απόδοση IOS για διάφορα λ .

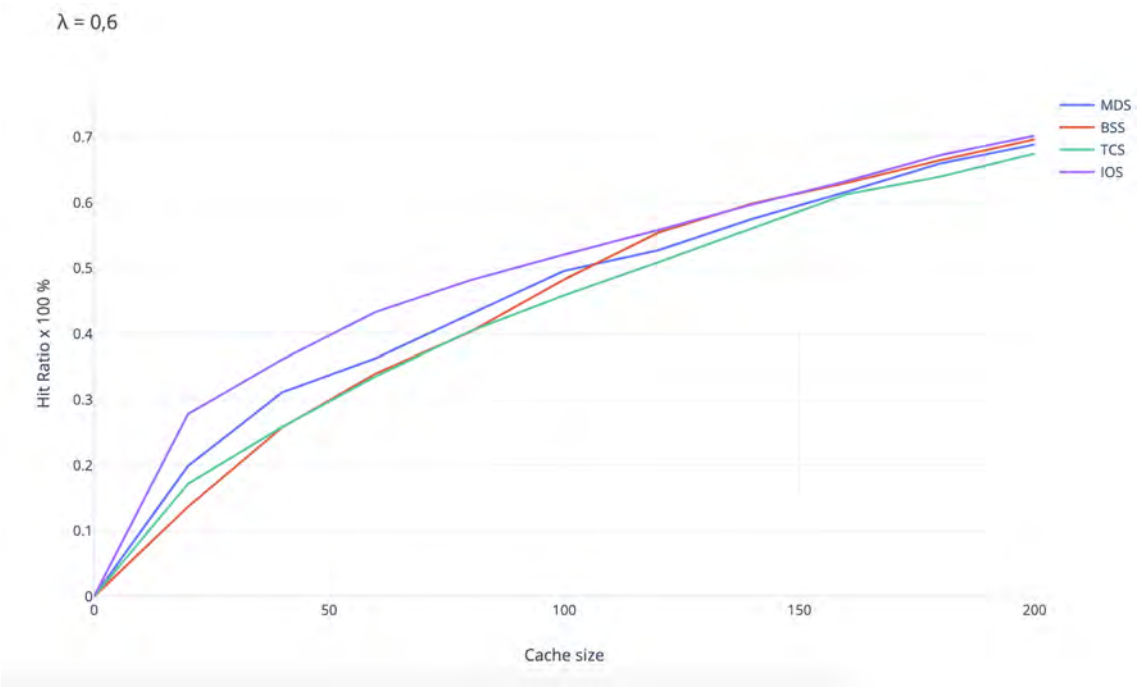
Τέλος, στο παραπάνω διάγραμμα παρουσιάζεται ο αλγόριθμος Insertion Order Significance. Ο συγκεκριμένος αλγόριθμος εμφανίζει για λ = 0,60 το μεγαλύτερο ποσοστό από κάθε άλλο αλγόριθμο. Το hit ratio για την παραπάνω περίπτωση είναι 0.7019 % . Επίσης με βάση τα αποτελέσματα παρατηρείται ότι εκτός της περίπτωσης όπου το λ ισούται με 0,40 , οι υπόλοιπες περιπτώσεις εμφανίζουν μεταξύ τους παρόμοια αποτελέσματα. Αυτό φαίνεται γραφικά, αφού η κόκκινη, η μπλε και η μωβ καμπύλη σχεδόν εφάπτονται για τα διάφορα μεγέθη της cache.

Βλέπουμε πως και στους 4 αλγόριθμους που αναλύθηκαν παραπάνω, για μικρά λ παρουσιάζεται καλύτερη απόδοση στο hit ratio όταν το συνολικό μέγεθος της μνήμης είναι αρκετα μικρό. Πιο συγκεκριμένα, όταν το λ είναι μικρότερο από 0,5 το hit ratio φτάνει και στο 0.3% . Από τα πειράματα προκύπτει η παρακάτω εξηγηση:
Σε μικρές μνήμες 50 θέσεων και κάτω, όταν θέτουμε το λ μικρότερο του 0,5 , δεσμεύουμε παραπάνω θέσεις για τα πιο δημοφιλή blocks. Έτσι όταν ζητηθεί ένα block πολλές φορές, αυτό ήδη υπάρχει στο κύριο τμήμα της μνήμης. Στις περιπτώσεις όπου δεσμεύομαι παραπάνω θέσεις για τα λιγότερο δημοφιλή blocks (λ μεγαλύτερο από 0,5) , εμφανίζονται περισσότερες αστοχίες στη μνήμη διότι τα δημοφιλη blocks που αποθηκεύονται είναι λιγότερα. Σε περιπτώσεις όπου το συνολικό μέγεθος της μνήμης είναι 50 θέσεις και άνω, αυτή η συμπεριφορά της μνήμης ως προς τα hits αρχίζει και εξαλείφεται, αφού ήδη υπάρχουν περισσότερες θέσεις στη μνήμη για να κρατήσουν ακόμη και τα όχι τόσο σημαντικά blocks.

6.2.2 Σύγκριση αλγορίθμων

Σύμφωνα με τα παραπάνω αποτελέσματα, παρατηρείται πως η βέλτιστη κατανομή του συνολικού μεγέθους της μνήμης στα τμήματα που την αποτελούν δίνεται όταν η τιμή της μεταβλητής λ ισούται με 0,60. Για παράδειγμα, όταν η cache είναι 200 θέσεων, το τμήμα της μνήμης που είναι υπεύθυνο για τα μη δημοφιλή blocks (early part) έχει μέγεθος 120 θέσεις και οι υπόλοιπες 80 είναι το μέγεθος του τμήματος της cache όπου φιλοξενεί τα πιο δημοφιλή blocks.

Στην παρακάτω εικόνα, παρουσιάζεται η απόδοση όλων των αλγορίθμων για λ = 0,6 .



Σχήμα 6.5: Απόδοση αλγορίθμων - Συγκεντρωτικός.

Από το διάγραμμα της παραπάνω εικόνας παρατηρούμε πως τα μεγαλύτερα ποσοστά hit ratio εμφανίζονται κατά φθίνουσα σειρά ως εξής:

- **IOS** (max hit ratio: 0.7019 %)
- **BSS** (max hit ratio : 0.6963 %)
- **MDS** (max hit ratio : 0.6886 %)
- **TCS** (max hit ratio : 0.6748 %)

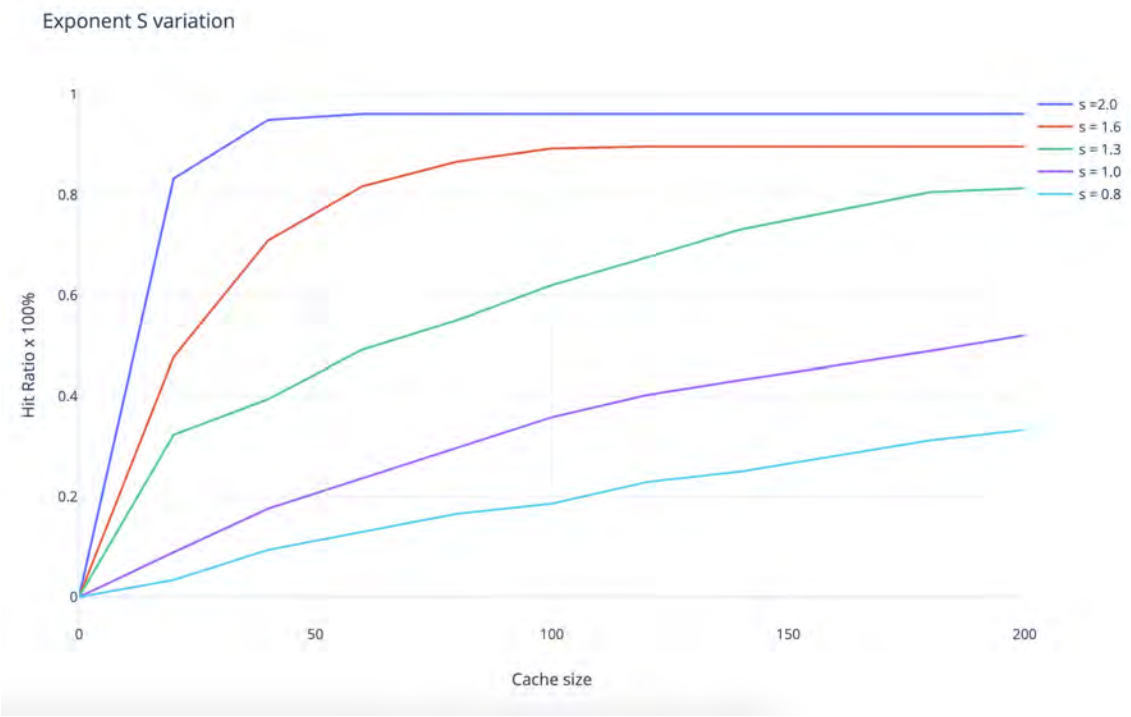
6.2.3 Πολλαπλά data set

Τα προηγούμενα πειράματα εκτελέστηκαν για ένα σύνολο δεδομένων που υπακούουν στην κατανομή zipf. Όπως είδαμε στο προηγούμενο κεφάλαιο, για τον εκθέτη του νόμου zipf που παρήγαγε τα δεδομένα χρησιμοποιήθηκε η τιμή s = 1,1 .

Παρόλα αυτά, για τις ανάγκες των πειραμάτων, παράχθηκαν περισσότερα data sets για διάφορες τιμές του s. Όσο μικρότερη είναι η τιμή του s, τόσο πιο τυχαία είναι τα δεδομένα / blocks που παράγονται.

Στο παρακάτω γράφημα επιλέχθηκε να αναλυθεί το πως συμπεριφέρεται ο αλγόριθμος MDS (Mining Difficulty Significance) για τις διάφορες τιμές του εκθέτη του δυναμόνόμου zipf. Πιο συγκεκριμένα εξετάζεται το πως αποδίδει ο παραπάνω αλγόριθμος ως προς το hit ratio της μνήμης cache για σταθερό λ = 0,6.

Οι τιμές του εκθέτη για την παραγωγή των δεδομένων των πειραμάτων κυμαίνονται από 0,8 (πιο τυχαία κατανομή block) εως 2,0 (συγκεκριμένα blocks εμφανίζονται πολύ συχνά).



Σχήμα 6.6: Παραλαγές εκθέτη s του zipf νόμου.

Παρατηρώντας το σχήμα 6.6 βλέπουμε, όπως ήταν αναμενόμενο, πως η απόδοση της μνήμης ως προς το hit ratio είναι πολύ χαμηλή όταν ο εκθέτης είναι κάτω από τη μονάδα. Η σχεδόν τυχαία κατανομή των blocks έχει ως αποτέλεσμα η συγκεκριμένη πολιτική αντικατάστασης να μην έχει καλά αποτελέσματα. Το μέγιστο ποσοστό των hits ανέρχεται στο 0.33 % .

Όσο η τιμή του εκθέτη αυξάνεται, τόσο πιο αποδοτική γίνται η συγκεκριμένη πολιτική. Αξιοσημείωτη είναι η συμπεριφορά του αλγορίθμου όταν ο εκθέτης παίρνει τιμές από 1.3 ως 2.0 . Σε αυτές τις περιπτώσεις βλέπουμε πως οι καμπύλες έχουν απότομη κλίση καθώς ακόμη και όταν το μέγεθος της μνήμης είναι πολύ μικρό (20 θέσεις), τα ποσοστά των hits κυμαίνονται από 0.3% εως και 0.8 % . Η παραπάνω συμπεριφορά εξηγείται από το γεγονός ότι υπάρχει μεγάλη ζήτηση για συγκεκριμένα μόνο blocks.

Τέλος, για τις περιπτώσεις όπου ο εκθέτης παίρνει τιμές μεγαλύτερες από 1.6 , παρατηρείται πως το μέγιστο hit ratio επιτυγχάνεται από πολύ νωρίς, για τα μικρά μεγέθη της μνήμης, και η καμπύλη γίνεται ευθεία γραμμή παράλληλη στον άξονα x. Σε αυτές τις περιπτώσεις το μέγιστο ποσοστό των hits αγγίζει το 0.96 % . Ο λόγος για τον οποίο η καμπύλη συμπεριφέρεται με αυτόν τον τρόπο είναι διότι τα block που ζητούνται κατά την διάρκεια του πειράματος έχουν αποθηκευτεί από πολύ νωρίς στη μνήμη. Το 0.04% αστοχίας της μνήμης αναφέρεται στα cold misses τα οποία είναι αναπόφευκτα. Ο παραπάνω όρος αναλύεται στην επόμενη ενότητα.

6.3 Σχόλια

6.3.1 Cold misses

Όταν η μνήμη cache είναι άδεια, στην αρχή του πειράματος, είναι λογικό να υπάρχουν misses μέχρι να αρχίσει να γεμίζει και να εμφανιστούν τα 0 πρώτα hits. Πιο συγκεκριμένα, όταν ένα block ζητείται για πρώτη φορά, αυτό θεωρείται miss για τη μνήμη. Με βάση αυτό, στα πειράματα που έγιναν το μέγιστο hit ratio που επιτυγχάνουν οι πολιτικές αντικατάστασης που υλοποιήθηκαν για την default τιμή του zipf εκθέτη είναι 0.737%. Αυτό σημαίνει πως το υπόλοιπο 0.263 % αντιστοιχεί στο ποσοστό των cold start misses ή αλλιώς first reference misses.

6.3.2 Στατιστικά δημοφιλίας των blocks

Ο διαχωρισμός της μνήμης σε τμήματα , καθιστά το κύριο τμήμα υπεύθυνο για την φιλοξενία των δημοφιλών blocks. Έτσι, μετά το τέλος των πειραμάτων, στο κύριο μέρος της μνήμης βρίσκονται blocks τα οποία περιλαμβάνουν και ένα αναγνωριστικό της δημοφιλίας τους. Για παράδειγμα , στα παραπάνω πειράματα, αποθηκεύονται blocks τα οποία έχουν popularity μέχρι και 169. Αυτό σημαίνει ότι ζητήθηκε 169 φορές, στοιχείο πολύ χρήσιμο για την cache.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική έρευνα

7.1 Συμπεράσματα

Οι πολιτικές αντικατάστασης σε μνήμες cache ήταν, είναι και θα είναι ένα πολύ σημαντικό θέμα για όλα τα συστήματα. Στην παρούσα εργασία τέθηκε ως στόχος η ανάπτυξη νέων πολιτικών αντικατάστασης οι οποίες βασίζονται στη σημαντικότητα των δεδομένων που αποθηκεύονται σε αυτήν.

Είδαμε πως για τις εφαρμογές που επεξεργάζονται και αποθηκεύουν δεδομένα από blockchain μπορούν να αναπτυχθούν τέτοιες πολιτικές αντικατάστασης οι οποίες αντί να αποθηκεύουν ολόκληρη τη δομή blockchain, φιλοξενούν στη μνήμη μόνο τα σημαντικά blocks από αυτό.

Τέλος, αναλύθηκε και ένα μοντέλο της μνήμης cache η οποία είναι χωρισμένη σε τμήματα, το καθένα από αυτά έχει τη δική του χρησιμότητα. Έτσι, μετα το τέλος των πειραμάτων, είδαμε πως στο κύριο τμήμα αποθηκεύονται τα πιό σημαντικά blocks για το σύστημα που προσομοιώθηκε.

7.2 Προτάσεις για μελλοντική έρευνα

Μια μελλοντική επέκταση της παρούσας εργασίας θα μπορούσε να είναι η χρησιμοποίηση πραγματικών blocks από blockchain με ρεαλιστικές τιμές και όχι η προσομοίωση αυτών. Σημαντικό είναι να αναφερθεί πως δεν λήφθηκαν υπόψη οι συναλλαγές του κάθε block, παρά μόνο ο αριθμός τους (Transaction Counter). Επίσης θα μπορούσαν να αναπτυχθούν και άλλοι αλγόριθμοι οι οποίοι, για το κάθε block, λαμβάνουν υπόψη τον δείκτη στο προηγούμενο block στο blockchain.

Αναφορές

[1] *Cache replacement policies*. URL: https://en.wikipedia.org/wiki/Cache_replacement_policies.

[2] *NodeJs*. URL: https://nodejs.org/en/.

[3] *LinkedHashMap*. URL: https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html.

[4] *Plot.ly*. URL: https://plot.ly/create/#/.

[5] *JSON*. URL: https://www.json.org/.

[6] *The Internet protocol journal*. URL: http://ipj.dreamhosters.com/wp-content/uploads/2017/12/ipj20-3.pdf.

[7] Henri Gilbert and Helena Handschuh. "Security analysis of SHA-256 and sisters". In: *International workshop on selected areas in cryptography*. Springer. 2003, pp. 175–193.

[8] *History of Bitcoin*. URL: https://en.wikipedia.org/wiki/History_of_bitcoin.

[9] *Power Law*. URL: https://en.wikipedia.org/wiki/Power_law.

[10] *Zipf Law*. URL: https://en.wikipedia.org/wiki/Zipf%5C%27s_law.

[11] *Power laws, Pareto distributions and Zipf's law*. URL: http://piketty.pse.ens.fr/files/powerlaws80-20rule.pdf.